ACAD Status Report
Deliverable D2

Version: 1.3, September 7, 2017

Erwin R. Komen
Radboud University Nijmegen

With the ACAD project having had its official start in May 2017, this "Deliverable D2" report details that status of the project so far. This report is meant to describe the finishing of ACAD's first work-package WP1, which consists of the components D1 and M1.

1) D1 – *start in M1, duration 0.5 PMs*
   Text preparation; identification of the selection of the corpora
2) M1 – *start in M2, duration 1 PM*
   Alpino conversion of new texts; description metadat in CMDI

A component that has officially been set to start later is already on its way too, so this status report will also inform the reader on the progress made with component S1.

3) S1 – *start in M4, duration 2 PMs*

This overview takes as a basis the project categories from the original CLARIAH Reearch Pilot Call Proposal. These categories are best illustrated by the project overview that is on page 4 of the original research proposal, and which is repeated here as Figure 1.

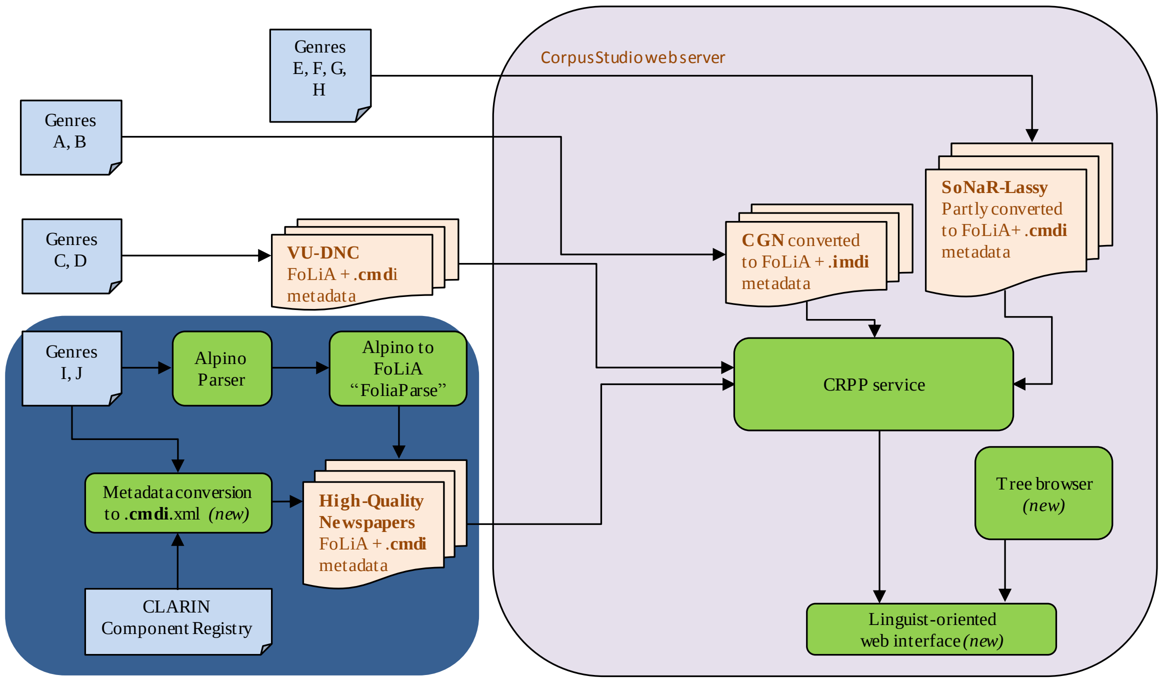
Genres E, F, G, H
CorpusStudio webserver
Genres A, B
SoNaR-Lassy Partly converted to FoLiA+ .cmdi metadata
CGN converted to FoLiA + .imdi metadata
Genres C, D
VU-DNC FoLiA + .cmdi metadata
Genres I, J
Alpino Parser
Alpino to FoLiA "FoliaParse"
CRPP service
Metadata conversion to .cmdi.xml (new)
High-Quality Newspapers FoLiA + .cmdi metadata
Tree browser (new)
CLARIN Component Registry
Linguist-oriented web interface (new)

Figure 1 Overview of the ACAD components

The deliverable D1 involves all components set out against the blue background of Figure 1. It starts with the box labelled "Genres I,J", which are two new genres that are to be added to the existing ones: (i) articles from the high-language newspaper "NRC", and (ii) conversations from WhatsApp.

Table 1 Status overview of all ACAD components

Component	Part	Item	Who	Status
Genres I, J	NRC to Folia		Micha	basis
	WhatsApp to Folia		Micha	*starting*
	CRPP integration		Erwin	-
VU-DNC to FoLiA	VU-DNC FoLiA		Erwin	*in process*
	CRPP integration		Erwin	-
Metadata Conversion	NRC metadata		Micha	*pending*
	DNC metadata		Micha	*in process*
	WhatsApp metadata		Micha	-
	Cesar integration		-	
	CRPP integration		Erwin	-
Linguist-Oriented Web Interface	Project Definition	1 ProjType	Erwin	done
		2 Search Elements	Erwin	done
		3 Global Vars	Erwin	done
		4 Data-dep vars	Erwin	done
		42 Var/SearchEl	Erwin	done
		43 Functions	Erwin	*started*
		44 Arguments	Erwin	*started*
		6 Conditions	Erwin	*started*
		7 Features	Erwin	-
	Search-Text Selector	*to be determined*	-	
	Project Execution	Xquery converter	Erwin	
		UI integration	-	
Tree browser	Text browser	Select Text	Erwin	done
		Show Text	Erwin	done
		Show Line Syntax	Erwin	done
	Result Database Viewer	*to be determined*	-	
		to be determined	-	

1 Adding new genres

Main idea: add genres 'NRC' and 'WhatsApp' to the inventory. This means the texts in these genres need to be syntactically processed and then their FoLiA counterparts need to be integrated into the CorpusStudio /crpp web service.

1.1 NRC newspaper

Almost all NRC newspaper articles that were due to be syntactically processed have been dealt with and are now available in FoLiA format. There is a small amount of about 20 texts on which the parser 'hangs'. We have decided to leave these texts out of the corpus for now. Should there be time left later in the project, we intend to investigate the reason for the problems and then we will try to add them.

1.1.1 The conversion process

It would be good (for posterity) to note here the steps we have taken to perform the conversion process of text to **.folia.xml** file.

#	Task	Program	Notes
1	Text to basic FoLiA	Frog	*Existing*
2	Add syntax layer to FoLiA	FoliaParse Alpino	Adapted own C# program *Existing*

We tried several ways, but ended up using a two-step method. The first step is to use the existing program 'Frog', which turns Dutch texts into **.folia.xml** files, retaining the paragraph structure. Frog adds several layers to the text, but it doesn't add the kind of syntax layer we are interested in. We use the existing program Alpino in step #2 through an intermediary C# program 'FoliaParse', in order to get the syntax layer. FoliaParse takes an existing FoLiA file, and line by line sends a sentence to the Alpino program (running on one of our servers). It then converts the returned *xml* data of the sentence's syntax to a syntax layer, and adds it into the **.folia.xml** file.

The other texts in the available corpora (SoNaR/Lassy-Groot, CGN) have all undergone a 'syntactic surfacing' process: the mainly hierarchically oriented dependency-like syntax trees have all been post-processed so as to arrive at trees where each sub-tree of a sentence *always* contains a *consecutive* piece of the text (Komen 2015).[1]

1.1.2 Notes

There is one other note that needs to be made here. Newspaper articles do not only consist of text, but they also contain titles, subtitles, figures, captions and so forth. The current conversion process has only treated the text, though we were able to include the paragraph structure of the text in it. Adding the other items (titles, captions) would require handwork. We have decided to not do this right now, given the small amount of time available for this process at all.

1.2 WhatsApp

The syntactic analysis and the conversion of the whatsapp messages into FoLiA format is still in the initial stages. We have reviewed the work done by a previous student-assistant on the texts, but the process of preparing the texts before the syntactic analysis can start is not finished yet. A small hickup, for example, is that fact that some of the dates for the apps do not include the year.

2 VU-DNC to FoLiA

The original idea was to take the files from the VU-DNC corpus and, if needed, convert them into FoLiA.

[1] That would not be the case, for instance, in a sentence like "*Ze probeert erachter te komen waar het geluid vandaan komt*". The Alpino parser sees "*erachter waar het geluid vandaan komt*" as one constituent, but this is not a constituent containing a consecutive piece of (surface) text, since the words "*te komen*" are missing: "*er achter te komen waar het geluid vandaan komt*". Depending on the syntactic search, the hierarchy or the surface constituent order may be of more interest to a researcher.

What has been done:

1) The files are already available in the .folia.xml format
2) The files already come with .cmid.xml metadata information
3) The whole VU-DNC archive has been put on an additional 'ACAD' disk of the CorpusStudioWeb VM (hosted by SurfSARA)

Actions remaining:

1) CRPP integration:
    a) unpack the files to the correct location on the ACAD disk
    b) add the location and information on these files to the CRPP back-end
    c) add the new information to the CESAR front-end

Additional problems that may come up:

1) The back-end CRPP needs to be moved to a different computer. This may add delays.

3 Metadata conversion

The status depends on the particular part we are talking about. The main idea is that metadata are either added (created) or taken from where they exist and integrated into the Cesar front-end system, as well as into the crpp back-end (where needed).

3.1 NRC metadata

Harvesting and processing the **metadata** of the texts is a task that remains to be done. We intend to ask the help of a student-assistant and of the researcher in this project for the manual part associated with the metadata gathering. The data will initially probably be gathered in an Excel sheet, so the second step, the conversion of these data from the Excel table to individual **.cmdi** files, also remains to be done.

Another matter, which we had not included in the project proposal, is the IPR for the newspaper articles. We now have adopted the plan to try and get the same kind of permission to use the NRC texts as the permission originally requested for newspaper articles that are part of the VU-DNC corpus.

3.2 VU-DNC metadata

This is already available, since it is part of the VU-DNC corpus. Only integration awaits.

3.3 Whatsapp metadata

The **metadata** for the app-corpus remain to be added.

3.4 Cesar integration

This is my task [EK], but this will wait until the linguist-oriented web interface version 1.0 is ready.

3.5 Cesar integration

This is my task [EK], but this will wait until the linguist-oriented web interface version 1.0 is ready.

4 Linguist-oriented web interface

Since this is one of the core components of the ACAD project, offering the most significant challenges, the start of its construction has been moved forward to the first month of the project, paralleling the work on the other components.

The main idea of the linguist-oriented web interface is to provide a way to describe a search for a particular word or a syntactic construction, to execute and monitor that search, and to be able to review the results in a user-friendly way.

4.1 Project definition

The first step in the linguistic search is to define and describe the search in the form of a "search project". Several steps are involved here, each needing quite some programming effort.

4.1.1 ProjType (1)

Provide metadata on the project: ready.

4.1.2 Search Elements (2)

Allow defining the main words that need to be searched: ready.

4.1.3 Global Vars (3)

Allow defining 'global' textual variables: ready.

4.1.4 Data-dependant vars (4)

Allow defining the names and descriptions of variables whose value can differ per search element: ready.

4.1.5 Var/SearchEl (42)

Allow defining the main category of a data-dependant variable. Is it fixed for a particular search element? Can it be copied from a global variable or another data-dependant variable? Or is it a function? READY

4.1.6 Functions (43)

Allow choosing the **main** function a variable uses: BASIC FUNCTIONALITY.
The User-Interface of this component (and the '44' one) is one of the things that will be developed further in the course of this ACAD project through the interaction with the Researcher(s) involved in ACAD.

4.1.7 Arguments (44)

Allow defining each argument of the main function, and then recursively allow defining sub-functions, sub-arguments and so on: ready in principle (but how does it work in practice?)

4.1.8 Conditions (6)

Define the conditions that make a search result a real hit: IN PROGRESS

4.1.9 Features (6)

Define the textual output features that should accompany each search hit: TODO

4.2 Search-text selector

This still needs thinking-through. The user will need to be able to identify which text or which collection of texts are going to be searched through.

This will be the first thing on the list once 4.1, the project definition, is available in version 1.

4.3 Project execution

Once a search project has been defined as per 4.1, it needs to be translated into a CRPX project file and this file then needs to be executed on the back end. There are a number of steps here:

4.3.1 Cesax project to Crpx

This step does not only include the creation (or editing) of a Crpx project, but it also involves translating all the linguist-oriented functions and variable definitions into the Xquery computer language.

4.3.2 User-interface integration

This it not too difficult. There are three items here:

- There needs to be a place for the user to indicate he/she wants the project to be 'run' (on texts selected using the 4.2 text-selector).
- The Cesar interface needs to monitor and show the progress of the project execution.
- There needs to be a 'ready' or an 'error' indication upon completion or when there is some kind of an error.

5 Tree browser

The component "Tree browser" is a bit of a shortened name for two functionalities that are required within the web application: (i) browsing through texts and sentences in general, and (ii) browsing through the results of a syntactic search project.

5.1 Text browser

All three planned functions are working:

1) Select a text (from among a multitude of texts)
2) Show the text (visualize the text in manageable pages)
3) Show line syntax (visualize the syntax tree of one selected line in a text)

5.2 Result database viewer

This component is still completely blank—it needs to be constructed and implemented.

The CESAR interface is actually only interested in the 'database' results that are produced by the CorpusStudio back-end. Processing and visualizing them is going to be something of a challenge, though.

1) Any result-database that is created upon execution of a search needs to become available to the Cesar front-end interface.
   a) Design/speed issue: copy the information to the existing Cesar (!) database or keep it separate? If the latter: how can it be accessed?

2) User interface to the results of a particular search:
   a) Provide a paginated 'list view' to allow the user to leaf through all the results
   b) Add sorting and filtering to the list view
   c) Provide a 'details view':
      i) Allow user to see all the features for one particular result
      ii) Allow user to *edit* database fields
   d) Provide an 'export' possibility: all the results should be made available in a CSV format so that they can be processed further in Excel or SPSS.

6 References

Komen, Erwin R. 2015. Surfacing Dutch syntactic parses. In *Computational Linguistics in the Netherlands (CLIN-26)*. Amsterdam.